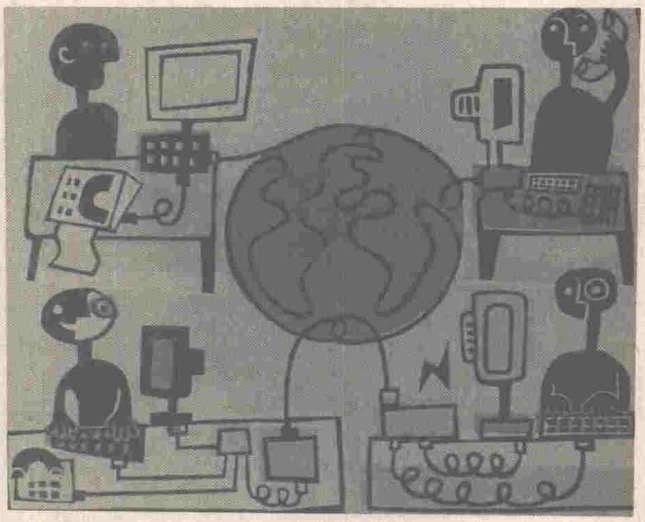
ภาษิต ภัทรมูล

รีเลชันแนล ภาคปฏิบัติการ

มองเห็นและเข้าใจในแนวคิด พร้อมทั้งหลักการของ รีเลชันแนลในระดับหนึ่ง



ถ้าจะศึกษารีเลชันแนลในแนวทฤษฎีจริง ๆ ที่ฝรั่งเขาเขียนให้อ่าน รับรองว่าเราจะต้องพบกับความยุ่งยากกับศัพท์แสงต่าง ๆ มากมาย แม้เป็นเรื่องเดียวกันบางทีก็มีคำเรียกหลายคำ บางครั้งก็พบกับนิยามที่อ่านแล้วทำความเข้าใจยาก... ยากที่จะวาดมโนภาพตามได้ คราวนี้เรามาลองเก็บตำรับตำราเหล่านั้นให้มิดชิด แล้วทำความเข้าใจใหม่ตามแนวทางของเรา ดูซิว่าจริง ๆ แล้วเรื่องนี้มันจะยากเกินความเข้าใจ และซับซ้อนเกินสามัญสำนึกของเรา ๆ ท่าน ๆ หรือไม่

วัตถุประสงค์

ผู้เขียนเพียงแต่ต้องการแสดงว่า บางครั้งทฤษฎีก็เป็นเรื่องของนักวิชาการที่จะศึกษาเพื่อแสดงถึงภูมิรู้อันเปล่งปลั่ง แต่การทำงานจริงเป็นอีกเรื่องหนึ่ง คนที่ทำงานจะรู้ว่าความเข้าใจอย่างแท้จริงคือแก่นแท้ที่ไม่จำเป็นต้องอาศัยเปลือกหรือความหรูหราภายนอกมาอ้างอิงแต่อย่างใดเลย

แต่การจะเข้าใจอะไรสักอย่าง ก็ไม่ใช่เรื่องง่ายดาย และไม่ใช่เกิดจากการลิขิตของฟ้าดินอย่างแน่นอน แต่เป็นเรื่องที่เราต้องทุ่มเทให้ความสนใจ หมั่นคิดและไตร่ตรองซ้ำแล้วซ้ำเล่า ให้กาลเวลาบ่มเพาะจนเกิดเป็นความกระจ่างแจ้งที่จะติดตัวเราไปชั่วชีวิต

แนวคิดหลักของรีเลชันแนล

แค่เพียงชื่อก็บอกอยู่โต้ง ๆ ว่าคือความสัมพันธ์ แนวคิดหลักและพื้นฐานจริง ๆ ของรีเลชันแนล จึงมาอยู่ที่ความสัมพันธ์ของข้อมูล ไม่ว่าเรากำลังทำระบบฐานข้อมูลจากโปรแกรมตัวใดอยู่ก็ตามเถอะ เรายังต้องเจอกับข้อมูลที่มีความสัมพันธ์กันอย่างหลีกเลี่ยงไม่ได้ แม้ความหมายทั้งหมดของรีเลชันแนลไม่ได้อยู่แค่นี้ แต่ตรงนี้ก็เป็นจุดเริ่มต้นที่ดี

ความสัมพันธ์ของข้อมูล?

ขอยกตัวอย่างข้อมูลที่ไม่มีความสัมพันธ์กันก่อน

ตารางที่ 1 เป็นตารางที่เก็บ รหัสประจำตัว ชื่อ-สกุล ที่อยู่ เบอร์โทรศัพท์ ของพนักงานขายในบริษัท ก.

รหัสประจำตัว	ชื่อ-สกุล	ที่อยู่	เบอร์โทรศัพท์
001	นายจำลอง ครองเมือง	20/1 ม.8 ต.นิเวศน์	886434
002	นายสมชาย ประกอบ	8 ม.6 ต.ในเมือง	564512
003	นายเพียร สอนนอก	7 ซอยร่วมใจ ต.ในเมือง	456121

ตารางที่ 1

ตารางที่ 2 เป็นตารางที่เก็บ เลขที่ใบสั่งซื้อสินค้า วันสั่งซื้อ มูลค่า ของบริษัท ก.

เลขที่ใบสั่งซื้อสินค้า	วันสั่งซื้อ	มูลค่า
000001	10/11/2541	8,000
000002	10/11/2541	5,200
000003	11/11/2541	8,000
000004	11/11/2541	7,800
000005	12/11/2541	5,400

ตารางที่ 2

จะเห็นว่าตารางทั้งสองไม่มีความสัมพันธ์ใด ๆ ต่อกัน เราไม่มีทางรู้เลยว่าใบสั่งซื้อแต่ละใบเป็นผลงานของพนักงานขายคนใด ข้อมูลที่มีอยู่จึงไม่มีประโยชน์ที่จะไปใช้งานอะไร นอกจากเก็บไว้คำนวณในตัวของมันเอง

คราวนี้เกิดเจ้าของบริษัท ก. อยากจะรู้ยอดขายของพนักงานขายแต่ละคน ถ้าจะเอาจากข้อมูลที่เก็บไว้ในสองตารางนี้ คำตอบคือมืดแปดด้านครับ เราจึงแก้ไขตารางที่ 2 ใหม่ โดยเพิ่มรหัสอ้างอิงถึงไปถึงรหัสประจำตัวของพนักงานขายด้วย ข้อมูลทั้งสองตารางจึงเกิดความสัมพันธ์ขึ้นมา

รหัสประจำตัว	ชื่อ-สกุล	ที่อยู่	เบอร์โทรศัพท์
001	นายจำลอง ครองเมือง	20/1 ม.8 ต.นิเวศน์	886434
002	นายสมชาย ประกอบ	8 ม.6 ต.ในเมือง	564512
003	นายเพียร สอนนอก	7 ซอยร่วมใจ ต.กระสัง	456121

เลขที่ใบสั่งซื้อสินค้า	วันสั่งซื้อ	มูลค่า	รหัสอ้างอิง
000001	10/11/2541	8,000	002
000002	10/11/2541	5,200	001
000003	11/11/2541	8,000	003
000004	11/11/2541	7,800	001
000005	12/11/2541	5,400	002

เมื่อตารางใบสั่งซื้อสินค้ามีส่วนอ้างถึงตารางพนักงานขาย เราจึงสามารถนำมาประมวลผล ในการแจงนับใบสั่งซื้อ หรือยอดสั่งซื้อของพนักงานขายแต่ละคน เช่น เราจะรู้ว่า นายจำลองซึ่งมีรหัสประจำตัว 001 มียอดสั่งซื้อเข้ามา 2 รายการ ตามเลขที่ใบสั่งซื้อ 000002 และ 000004 ในวันที่ 10/11/2541, 11/11/2541 ตามลำดับ มูลค่ารวมทั้งสิ้น 13,000 บาท

ข้อเท็จจริงตามสามัญสำนึก

โดยไม่ต้องใช้ทฤษฎีให้ยุ่งยาก พวกเราบางคนคงมองออกแล้วว่าทำไมเราจึงต้องกำหนดรหัสของพนักงานสั่งซื้อขึ้นมา ทำไมเราไม่อ้างอิงโดยใช้ชื่อ-สกุลของพนักงานขายซึ่งดูจะง่ายและประหยัดเนื้อที่ของตาราง ทำได้ครับ **ถ้าพนักงานขายของเราทุกคนไม่มีชื่อและนามสกุลซ้ำกัน** ถึงแม้เป็นกรณีที่เกิดขึ้นได้ยาก แต่ก็เป็นไปได้ (ในอนาคตบริษัท ก. อาจมีพนักงานขายเพิ่มเป็นหมื่นคน หรือแสนคน ก็อาจมีชื่อ-สกุลซ้ำกัน) ดังนั้นเราจึงต้องกำหนดรหัสประจำตัวพนักงานแต่ละคน **ไม่ให้ซ้ำกันอย่างเด็ดขาด** จุดนี้คงไม่ต้องอธิบายว่าจะเกิดความสับสนเพียงใดถ้ามีรหัสประจำตัวซ้ำกันได้ คอลัมน์รหัสประจำตัวนี้ตามทฤษฎีเขาเรียกว่า **คีย์หลัก (Primary Key)** และรหัสอ้างอิงในตารางที่ 2 คือ **คีย์นอก (Foreign Key)** โดยคีย์นอกสามารถเชื่อมกับตารางอื่น และมีค่าที่ซ้ำซ้อนกันได้

เราลองสังเกตในตารางที่ 2 เห็นอะไรเพิ่มขึ้น? ถูกแล้วครับ เลขที่ใบสั่งซื้อสินค้าสามารถเป็นคีย์หลักได้ โดยนำไปเชื่อมโยงกับรายการสินค้าที่ถูกสั่งซื้อ มีประโยชน์สำหรับการตัดสต็อกและเช็คยอดขายของรายการสินค้าแต่ละตัวของบริษัท ก.

ถ้าเราใช้ SQL เลือกข้อมูลที่มีความสัมพันธ์ขึ้นมาโชว์ที่หน้าจอ (จะกล่าวถึง SQL ในช่วงหลังของบทความ) ก็จะเกิดภาพคล้ายกับว่า เรามีตารางเพิ่มขึ้นมาอีก 1 ตาราง ตารางสมมตินี้เกิดจากความสัมพันธ์ของตารางที่มีอยู่ โดยเฉพาะถ้าเราเพิ่มเติม แก้ไข และลบข้อมูลที่ต้องการในตารางสมมติได้เลย นี่คือคุณสมบัติที่มีในทฤษฎีรีเลชันแนล!

ดังนั้นแม้เราจะแบ่งการเก็บข้อมูลออกเป็นหลายตาราง แต่ในตารางเหล่านั้นมีความสัมพันธ์กัน เราก็จะได้ความเป็นอันหนึ่งอันเดียวกันของระบบฐานข้อมูล ตรงนี้เป็นประโยชน์ของรีเลชันแนล

ความสมบูรณ์ของข้อมูล

แล้วก็มีเหตุการณ์บางอย่างเกิดขึ้น นายจำลอง ครองเมือง ลาออก! การจัดการของระบบฐานข้อมูลคือตัดชื่อนายจำลองออกจากตารางที่ 1 แล้วก็ต้องตามไปลบในตารางใบสั่งซื้อสินค้าที่มีส่วนอ้างอิงถึงนายจำลองด้วย เพราะถ้าปล่อยเอาไว้ ข้อมูลนั้นก็จะกลายเป็นขยะ เพราะเป็นใบสั่งซื้อที่ไม่มีตัวตนผู้สั่งซื้อ (นายจำลอง หายไปจากตารางที่ 1 แล้ว)

นั่นคือ รีเลชันแนลจะต้องสามารถรักษาความสมบูรณ์ของระบบฐานข้อมูล โดยกระทำจากความสัมพันธ์ของข้อมูลเหล่านั้น

ทีนี้ถ้าเราไปลบใบสั่งซื้อที่เกี่ยวข้องกับนายจำลองในตารางที่ 2 โดยที่ยังมีชื่อนายจำลองอยู่ในตารางที่ 1 ความสมบูรณ์ของฐานข้อมูลจะเสียไปหรือไม่ คำตอบคือ ไม่เสียความสมบูรณ์ เพราะเป็นไปได้ว่าพนักงานขายคนนั้นอาจไม่มียอดสั่งซื้อเลยในช่วงเวลาหนึ่ง (โดยเฉพาะในภาวะเศรษฐกิจเช่นนี้!) และยอดสั่งซื้อของนายจำลองก็จะกลายเป็นศูนย์ ตรงนี้เป็นความแตกต่างที่ชัดเจนของคีย์หลักและคีย์นอก นั่นคือคีย์หลักจะมีบทบาทควบคุมคีย์นอก เปรียบเสมือนต้นไม้ใหญ่ เมื่อโค่นล้มไปเหล่านกกาและวานรก็อยู่ไม่ได้ แต่ถ้านกกาและวานรบางตัวจะอพยพจากต้นไม้ใหญ่ ก็ไม่มีอะไรกระทบกระเทือน

ความปลอดภัยของระบบฐานข้อมูล

เราคิดต่อไปว่า กรณีการไปลบใบสั่งซื้อของนายจำลองเกิดขึ้นโดยนายสมชาย เพราะเป็นคู่แข่ง และไม่ค่อยถูกกัน ก็ไม่เป็นการยุติธรรมกับนายจำลองนัก คนที่มีสิทธิ์ลบน่าจะเป็นผู้ที่มีอำนาจ เช่น เจ้าของบริษัท ก. หรือผู้ที่ได้รับมอบหมายอำนาจ

หลักการอีกอย่างของรีเลชันแนลคือต้องมีการรักษาความปลอดภัยของข้อมูล การกำหนดสิทธิและรหัสผ่านของผู้ใช้แต่ละคน เพื่อให้การใช้ระบบฐานข้อมูลมีความถูกต้อง เป็นจริงอยู่เสมอ และดำเนินไปอย่างเรียบร้อยตามกฎกติกา โดยถูกกำหนดจากโครงสร้างฐานข้อมูล ความสมบูรณ์ของฐานข้อมูล และจากความจำเป็นของผู้ใช้ซึ่งมีระดับแตกต่างกัน โดยมีข้อถือสาสำคัญที่สุดคือ ต้องรักษาความสอดคล้องของระบบฐานข้อมูลไว้ให้จงได้

มาถึงจุดชมวิวกลางทาง

มาถึงตรงนี้ เชื่อว่าพวกเราคงจะพอมองเห็นและเข้าใจในแนวคิดของรีเลชันแนลในระดับหนึ่ง ซึ่งจะตรงกับกฎหรือทฤษฎีทุกอย่างคงเป็นไปไม่ได้ ทั้งนี้ย่อมขึ้นอยู่กับตัวพัฒนาตัวฐานข้อมูลที่เราใช้อยู่ว่าจะมีส่วนสนับสนุนการรีเลชันแนลมากน้อยเพียงไร เราน่าจะเน้นที่แนวคิดหรือประโยชน์ของมันมากกว่าจะไปยึดถือตามทฤษฎีทุกสิ่งทุกอย่าง

ปฏิบัติการทางรีเลชันแนล

ส่วนสำคัญที่เราจะสร้างและใช้งานระบบฐานข้อมูลก็คือภาษาที่สั่งการ รีเลชันแนลก็เช่นกัน ภาษาที่ใช้ในการรีเลชันแนล ส่วนมากจะมีพื้นฐานมาจากรีเลชันแนลแคลคูลัส หรือพีชคณิตรีเลชันแนล ความแตกต่างคือ รีเลชันแนลแคลคูลัส ไม่เชิงจะเป็นภาษาที่มีแบบแผน แต่เป็นการใช้สัญลักษณ์ทางลอจิกในการแสดงถึงสูตรที่จะใช้ในการสอบถามข้อมูลกับระบบฐานข้อมูล พีชคณิตรีเลชันแนลมีโครงสร้างทางภาษาใกล้เคียงภาษามนุษย์ซึ่งง่ายกว่าจึงได้รับความนิยม ภาษาหลักที่เป็นพระเอกในปัจจุบันเห็นจะเป็น SQL (Structure Query Language) ก็มีพื้นฐานมาจากพีชคณิตรีเลชันแนล

SQL

เป็นภาษาที่ได้รับความนิยมสูงสุด โดยใช้การบรรยายในการทำงานกับระบบฐานข้อมูล การใช้งาน SQL ถูกรวมอยู่ตัวพัฒนาฐานข้อมูลประเภทต่าง ๆ เช่น C, Cobol, Pascal, Basic... จนมาถึงตัวพัฒนาฐานข้อมูลโดยตรงเช่น FoxPro, PowerBuilder, Oracle, Access เป็นต้น ซึ่งภาษา SQL ในตัวพัฒนาแต่ละตัวจะแตกต่างกันไปบ้างตามข้อกำหนดและภาษาที่ใช้ ในตัวอย่างต่อไปนี้ผู้เขียนจึงขออนุญาตใช้ SQL มาตรฐานเป็นหลัก เพราะถ้าไปอิงตัวใดตัวหนึ่งก็เกรงใจผู้ที่ไม่ได้ใช้เหมือนกัน และขอเลือกที่ใช้บ่อย ๆ โดยการใช้งานอย่างละเอียด เราสามารถไปหาอ่านได้ตาม Help และหนังสือคู่มือเกี่ยวกับ SQL ในท้องตลาด

การปฏิบัติการรีเลชันแนล

สามารถแบ่งตามหัวข้อหลัก ๆ ได้ดังนี้

1. การทำงานเกี่ยวกับตาราง เช่น สร้าง/แก้ไข/ลบตาราง ล็อกตาราง ปล่อยล็อกตาราง

2. การทำงานเกี่ยวกับข้อมูล เช่น เลือก เพิ่ม ลบ แก้ไข ข้อมูลในระเบียน

การทำงานเกี่ยวกับตาราง

1. การสร้างตาราง

- CREATE TABLE ใช้นิยามและสร้างตารางข้อมูลที่ต้องการ

ไวยากรณ์

```
CREATE TABLE [ชื่อตาราง]
([ชื่อคอลัมน์ที่ 1] [ชนิดข้อมูล][ขนาด][กฎข้อบังคับ],
[ชื่อคอลัมน์ที่ 2] [ชนิดข้อมูล][ขนาด][กฎข้อบังคับ],
[ชื่อคอลัมน์ที่ 3] [ชนิดข้อมูล][ขนาด][กฎข้อบังคับ],
.....
[ชื่อคอลัมน์ที่ N] [ชนิดข้อมูล][ขนาด][กฎข้อบังคับ])
```

ค่าที่สังเกต

```
ชนิดของข้อมูล (การอ้างชื่อจะแตกต่างกันในตัวพัฒนาแต่ละตัว)
กฎข้อบังคับ เช่น ยอมให้มีค่าว่างหรือไม่ หรืออื่น ๆ เช่น NULL, Not NULL,
UNIQUE, Check
```

ตัวอย่างการใช้งาน

```
CREATE TABLE employ
(code   char(8) Not NULL,
name    char(30),
address char(50),
phone   numberic(8))
```

ในตัวอย่างนี้ เราจะสร้างตารางใหม่ ชื่อว่า Employ โดยมีคอลัมน์หรือฟิลด์จำนวน 4 คอลัมน์คือ code, name, address และ Phone โดยมีชนิดข้อมูลและข้อบังคับตามอ้างไว้

2. แก้ไขตาราง

- ALTER TABLE ใช้แก้ไขโดยเพิ่มคอลัมน์ในตาราง

ไวยากรณ์

```
ALTER TABLE [ชื่อตาราง] add ([ชื่อคอลัมน์] [ชนิดของข้อมูล] [ข้อบังคับ])
```

ตัวอย่างการใช้งาน

```
ALTER TABLE employ add (address2 char(30))
```

คอลัมน์ Address2 ซึ่งมีชนิดเป็น Character ถูกเพิ่มเข้าในตาราง Employ

3. ลบตาราง

- DROP TABLE ใช้ลบหรือยกเลิกตารางออกจากระบบ

ไวยากรณ์

```
DROP TABLE [ชื่อตาราง]
```

ตัวอย่างการใช้งาน

```
DROP TABLE employ
```

คำสั่งเหล่านี้ค่อนข้างจะทำความเข้าใจได้ง่าย จุดสำคัญและลักษณะเด่นของ SQL จะอยู่ที่การดำเนินการกับข้อมูล การเลือกและประมวลผลเกี่ยวกับข้อมูลเสียมากกว่า ผู้เขียนจึงขอเน้นเป็นพิเศษสำหรับข้อที่ 2

2. การกระทำกับข้อมูล

2.1 การเลือกข้อมูล

- SELECT เป็นคำสั่งที่กระทำการแบบพีชคณิตรีเลชันแนลเกี่ยวกับการเลือก การเข้าถึงข้อมูลโดยตรง การ join จะทำการค้นหารายการจากตารางตั้งแต่หนึ่งตารางขึ้นไปตามเงื่อนไขที่สั่ง ผลลัพธ์ที่ได้จะเป็นเซตของข้อมูลที่สามารถสร้างเป็นตารางใหม่ หรืออาจจะใช้แสดงออกมาทางจอภาพเท่านั้น

ไวยากรณ์

```
SELECT [ALLIDISTINCT] select-list (รายละเอียดที่เลือก)
FROM ชื่อตาราง[ชื่อวิว] [ชื่อย่อ] (ตารางแหล่งที่มา)
[WHERE เงื่อนไข] (กำหนดเงื่อนไขของข้อมูลที่เลือก)
[GROUP BY ชื่อคอลัมน์ I หมายเลขตำแหน่งคอลัมน์ ] (การจัดกลุ่ม)
[HAVING เงื่อนไข] (กฎของการแสดงกลุ่ม)
[INTERSECTIMINUSIUNION [ALL] คำสั่ง Select] (การสอบถามย่อย)
[ORDER BY ชื่อคอลัมน์ I หมายเลขตำแหน่งคอลัมน์] (การจัดกลุ่ม)
[ASCIDESC] (การเรียงลำดับ)
[INTO ส่งผลลัพธ์ไปที่]
```

ตัวอย่างการใช้งาน

การเลือกข้อมูลในตารางเดี่ยว

ตัวอย่างที่ 1

```
SELECT ALL FROM employ WHARE code='001'
```

คำสั่ง SELECT จะเริ่มโดยการระบุรายละเอียดของสิ่งที่ต้องการ ALL คือเลือกทุกคอลัมน์ จาก FROM ต่อไปเราก็มองว่าเราจะไปดึงข้อมูลมาจากตารางไหน ในที่นี้คือ Employ (โดยเราสามารถระบุตารางมากกว่าหนึ่งตารางได้) WHERE เป็นการระบุเงื่อนไขว่าเลือกมาเฉพาะข้อมูลที่มีคอลัมน์ Code มีค่าเท่ากับ 001 การกระทำดังกล่าวจะเป็นไปโดยอัตโนมัติ เราไม่ต้องเปิดตารางขึ้นมาก่อนแต่อย่างใด

```
2. SELECT ALL FROM employ
```

จะแสดงรายการของตาราง employ ขึ้นมาทั้งหมด

```
3. SELECT code,name FROM employ
```

จะแสดงข้อมูลในคอลัมน์ code และ name ของตาราง employ

4. ให้ตาราง PROVINCE เป็นที่เก็บข้อมูลจังหวัดทุกจังหวัดในประเทศไทย โดยมีคอลัมน์ name เก็บชื่อจังหวัด

การเลือกแสดงข้อมูลรายชื่อทุกจังหวัดในประเทศไทยจึงเป็น

```
SELECT name FROM province
```

แต่ถ้ามีการคีย์ข้อมูลผิดพลาด จังหวัดอ่างทองอาจมีสองรายการการขึ้นโชว์ก็จะผิดพลาด ดังนั้นเราจึงเพิ่มคำสั่ง DISTINCT ลงไป

```
SELECT DISTINCT name FROM province
```

คำสั่ง DISTINCT จะช่วยเช็คตัวซ้ำและนำออกแสดงเพียงครั้งเดียว

5. ประโยคย่อย WHERE เราสามารถระบุเงื่อนไขได้โดยโอเปอเรเตอร์ทั่วไป เช่น NOT <> = กรณีค่าคอลัมน์เป็นตัวเลข เราก็สามารถระบุเงื่อนไขที่เป็นการคำนวณเช่น + - * /

ตัวอย่าง

```
SELECT ALL FROM  invoice WHERE (valueprice/2)>5000
```

6. คำว่า LIKE จะใช้กับค่าในคอลัมน์ประเภทตัวอักษร ว่าตรงกับประโยคที่ระบุหรือไม่ ใช้ร่วมกับ NOT จะเป็นตรงกันข้าม สามารถใช้เครื่องหมาย Widecard (เช่น * , ??,%) ในประโยคได้ (ขึ้นอยู่กับตัวพัฒนาที่ใช้ด้วย)

ตัวอย่าง

```
SELECT name FROM employ WHERE name LIKE 'ส%'
```

จะเป็นการเลือกพนักงานขายที่มีชื่อนำหน้าด้วยตัวอักษร "ส" ขึ้นมาแสดงทั้งหมด เราสามารถแทรกเครื่องหมาย Widecard ไว้ช่วงต้น ช่วงกลาง ช่วงท้ายประโยค ได้ตามความต้องการและความจำเป็น

7. เราสามารถใช้เงื่อนไขทางตรรกะ เช่น AND OR ในประโยค WHERE

ตัวอย่าง

```
SELETE code,name FROM employ WHERE code='001' OR code='002'
```

8. กรณีมีเงื่อนไขย่อยมากขึ้น ใช้คำ IN ร่วมในประโยค WHERE ได้ เช่น

```
SELECT code,name FROM employ WHERE code IN('001','002','003')
```

9. ในด้านการคำนวณ มีฟังก์ชัน COUNT, SUM, AVG, MIN, MAX ให้เราเลือกใช้

ตัวอย่าง

```
SELETE COUNT(code) FROM employ
```

จะแสดงจำนวนแจงนับของคอลัมน์ code ซึ่งก็คือจำนวนรายการทั้งหมดในตารางนั่นเอง

10. การนำผลการปฏิบัติการ SQL ไปเก็บในตาราง แบ่งออกเป็น 2 กรณีคือ

ก. ประโยค INTO DBF ชื่อตาราง จะนำไปเก็บในตารางซึ่งถูกสร้างขึ้นอัตโนมัติเป็นการถาวร

ข. ประโยค INTO CURSOR ชื่อตาราง จะนำไปเก็บในตารางชั่วคราว จะลบเมื่อมีการปิด

ตัวอย่าง

```
SELECT code,name FROM employ INTO CURSOR emtmp
```

11. เราสามารถส่งผลลัพธ์ไปพิมพ์หรือเก็บไว้ใน Text File โดยตรงด้วยคำสั่ง TO FILE ชื่อไฟล์ หรือ TO PRINTER

ตัวอย่าง

```
SELECT code,name FROM empoly TO PRINTER PLAIN
```

PLAIN เป็นการระบุไม่ได้พิมพ์ส่วนหัว (ชื่อคอลัมน์)

12. GROUP BY และ HAVING คำสั่ง SELETE จะแสดงในทุก ๆ ระเบียนที่ตรงตามเงื่อนไข แต่ถ้าเราต้องการการจัดกลุ่มในรายการที่ SELECT มาแล้วโดยจัดเป็นกลุ่มให้เหมือนว่า ในแต่ละกลุ่มนั้นแทนรายการเดียวทำได้โดยการใช้ GROUP BY

ตัวอย่าง

กำหนดให้ตาราง invioce ประกอบด้วยคอลัมน์ repcode (รหัสอ้างอิงพนักงานขาย), no_inv (หมายเลขใบสั่งซื้อ), day_inv (วันสั่งซื้อ) และ valueprice (มูลค่าการสั่งซื้อ)

เราต้องการหาค่าของยอดจำหน่ายของพนักงานขายแต่ละคน โดยจำแนกตาม repcode ยอดรวมจะอยู่ในคอลัมน์ สมมติชื่อ total ก็สามารถเขียนเป็น SQL ได้ดังนี้

```
SELECT repcode,SUM(valueprice) AS total FROM invoice GROUP BY repcode
```

ประโยค HAVING นั้นจะเป็นตัวเพิ่มเงื่อนไขให้แก่ GROUP ORDER ในตัวอย่างถ้าเราไม่ต้องการให้โชว์พนักงานขายที่มียอดจำหน่ายรวมเป็น 0 สามารถเขียนเป็นประโยค SQL ได้ดังนี้

```
SELECT repcode,SUM(valueprice) AS total FROM invoice GROUP BY repcode
HAVING SUM(valueprice)>0
```

การเลือกข้อมูลในตารางตั้งแต่ 2 ตารางขึ้นไป

เราสามารถใช้คำสั่งและเงื่อนไขต่าง ๆ ได้เช่นเดียวกับการทำงานกับตารางเดี่ยว แต่ก็มีส่วนที่น่าสนใจควรจะกล่าวถึงดังนี้

1. การสอบถามโดยอาศัยความสัมพันธ์ระหว่างตาราง

ตัวอย่างนี้ให้ท่านดูตารางที่ 1 และตารางที่ 2 (ที่เพิ่มความสัมพันธ์แล้ว) ขออนุญาตดึงมาแสดงตรงนี้เลย

รหัสประจำตัว	ชื่อ-สกุล	ที่อยู่	เบอร์โทรศัพท์
001	นายจำลอง ครองเมือง	20/1 ม.8 ต.นิเวศน์	886434
002	นายสมชาย ประกอบ	8 ม.6 ต.ในเมือง	564512
003	นายเพียร สอนนอก	7 ซอยร่วมใจ ต.กระสัง	456121

เลขที่ใบสั่งซื้อสินค้า	วันสั่งซื้อ	มูลค่า	รหัสอ้างอิง
000001	10/11/2541	8,000	002
000002	10/11/2541	5,200	001
000003	11/11/2541	8,000	003
000004	11/11/2541	7,800	001
000005	12/11/2541	5,400	002

ตารางที่ 1 ข้อมูลพนักงานขาย มีชื่อว่า employ

กำหนดให้มีคอลัมน์ เลขประจำตัว (code), ชื่อ-สกุล (name), ที่อยู่ (address), เบอร์โทรศัพท์ (phone)

ตารางที่ 2 ข้อมูลใบสั่งซื้อ มีชื่อว่า invoice

กำหนดให้มีคอลัมน์ เลขที่ใบสั่งซื้อ (no_inv), วันที่สั่งซื้อ (day_inv), valueprice (มูลค่า), รหัสอ้างอิง (repcode)

เราต้องการดูข้อมูลใบสั่งซื้อซึ่งขึ้นมาพร้อมกับเลขประจำตัวและชื่อพนักงานขาย สามารถเขียนประโยค SQL เป็น

```
SELECT employ.code, employ.name ,invoice.no_inv, invoice.day_inv,
invioce.valueprice FROM employ,invoice WHERE employ.code=invoice.repcode
```

ซึ่งข้อมูลก็จะขึ้นมาแสดงดังตารางนี้

รหัสประจำตัว	ชื่อ-สกุล	เลขที่ใบสั่งซื้อสินค้า	วันสั่งซื้อ	มูลค่า
002	นายสมชาย ประกอบ	000001	10/11/2541	8,000
001	นายจำลอง ครองเมือง	000002	10/11/2541	5,200
003	นายเพียร สอนนอก	000003	11/11/2541	8,000
001	นายจำลอง ครองเมือง	000004	11/11/2541	7,800
002	นายสมชาย ประกอบ	000005	12/11/2541	5,400

จะเห็นว่า SQL ช่วยให้เราสามารถใช้ประโยชน์จากข้อมูลที่มีความสัมพันธ์กัน โดยการเขียนคำสั่งเพียงเล็กน้อยเท่านั้น โดยที่เราไม่ต้องไปกำหนดคำสั่งการเชื่อมโยงอื่นใด การทำงานจะเป็นไปโดยอัตโนมัติ เพราะ SQL เป็นภาษาที่ถูกสร้างขึ้นมาเพื่อจัดการกับความสัมพันธ์ของข้อมูลโดยตรงอยู่แล้ว

2. คำสั่ง SELECT สามารถใช้ซ้อนกันได้ ตรงจุดนี้บรรดาเซียน MS.ACCESS คงคุ้นเคยและใช้กันอยู่ประจำ เพียงแต่การทำงานในลักษณะวิชวลทำให้ทุกอย่างดูง่ายจนไม่รู้สึกถึงความซับซ้อนของมันเท่านั้น นั่นคือ Query ซ้อน Query ไงครับ

สมมติว่าเราต้องการสกรีนข้อมูลจากตารางที่แสดงไปก่อนหน้านี้ ให้แสดงเฉพาะใบสั่งซื้อที่ไม่ลงชื่อพนักงานขาย เราก็สามารถนำประโยค SQL เดิมมาใช้ได้เลย

```
SELECT ALL FROM  invoice WHERE repcode NOT IN
(SELECT employ.code ,employ.name, invoice.no_inv, invoice.day_inv,
invioce.valueprice FROM employ, invoice WHERE
employ.code=invoice.repcode)
```

3. ประโยค INTERSECT, UNION, MINUS จะเป็นการทำงานในลักษณะของเซตระหว่างสองตาราง ความทรงจำอันเลือนลางเกี่ยวกับเซตตั้งแต่สมัยเรียนมัธยมบอกเราว่า

```
INTERSECT = คัดเลือกรายการที่เหมือนกันในตารางสองตาราง โดยที่
ตัดรายการซ้ำออกให้เหลือเพียงรายการเดียว (ตามความเป็นจริงยังไม่พบว่ามี
การใช้ในตัวระบบฐานข้อมูลใด)
UNION = ผนวกรายการจากตารางสองตาราง โดยที่ตัดรายการซ้ำออก
ให้เหลือเพียงรายการเดียว
MINUS = ทำ INTERSECT แล้วเลือกเอาเฉพาะแต่รายการที่ไม่ถูก
INTERSEC ของตารางที่ MINUS (ยังไม่พบการใช้งานจริงเช่นเดียวกับ
INTERSECT)
```

สรุปว่า คำสั่ง SELECT เป็นคำสั่งซึ่งมีความซับซ้อนพอสมควร ต้องอาศัยการทำความเข้าใจและฝึกฝน ในความซับซ้อนนั้นไม่ได้ยากอย่างที่หลายคนเบือนหน้าหนี เพราะ SQL เป็นภาษาที่มีรูปแบบจำเพาะตายตัว ความซับซ้อนจึงน่าจะอยู่ที่การใช้เงื่อนไขทำงานกับข้อมูล ซึ่งโปรแกรมเมอร์ทั่วไปก็มีความถนัดในแง่การใช้ตรรกะอยู่แล้ว

2.2 การเพิ่มข้อมูลในตาราง

- INSRET INTO ใช้แทรกข้อมูลมากกว่าหนึ่งแถวในตาราง หรือวิวที่กำหนด

ไวยากรณ์

```
INSERT INTO [ชื่อตารางชื่อวิว]
[ชื่อคอลัมน์1,คอลัมน์2,....]
[VALUES (ค่าคงที่ตัวแปร|NULL) (ที่จะใส่ไปในคอลัมน์ 1, 2 ...ตามลำดับ)
[คำสั่ง SELECT]
```

ข้อสังเกต

ถ้าระบุเงื่อนไข VALUES จะเพิ่มเพียงหนึ่งรายการ แต่ถ้าใช้ร่วมกับ SELECT จะเพิ่มได้มากกว่าหนึ่งรายการ

ตัวอย่าง

```
INSERT INTO employ VALUES(code,name,address,phone)
VALUES('004','นายสมปอง สงวนรัมย์','141/1 ม.5 ต.สนามชัย',658471)
```

จะเพิ่มรายการในตาราง employ อีกหนึ่งรายการ

```
INSERT INTO tmpemb
SELECT ALL FROM employ
```

จะเพิ่มรายการในตาราง tmpemb จากข้อมูลทุกรายการในตาราง employ

2.3 การลบข้อมูลในตาราง

- DELETE FROM ใช้ลบรายการข้อมูลออกจากตาราง

ไวยากรณ์

```
DELETE FROM [ชื่อตารางชื่อตารางวิว]
WHERE CURRENT OF ชื่อเคอร์เซอร์
```

ตัวอย่าง

```
DELETE FROM employ WHERE CURRENT OF tmpemb
```

ข้อสังเกต

ก่อนเรียกใช้คำสั่งนี้จะต้องมีการสร้างเคอร์เซอร์ และเปิดใช้เคอร์เซอร์เสียก่อน

2.4 การแก้ไขข้อมูลในตาราง

- UPDATE ใช้ในการแก้ไขข้อมูลในตาราง โดยแก้ในคอลัมน์ที่มีค่าตรงตามเงื่อนไข

ไวยากรณ์

```
UPDATE [ชื่อตารางชื่อวิว]
SET [ชื่อคอลัมน์=ค่าที่จะใส่เข้าไปในคอลัมน์นั้น ๆ]
[WHERE          ประโยคเงื่อนไข]
```

ตัวอย่าง

```
UPDATE employ
SET PHONE='123456'
WHERE  code='004'
```

เติมอย่างไรก็ไม่เต็ม

ภาษา SQL ขั้นพื้นฐาน (แต่เป็นส่วนสำคัญที่ใช้หากินได้) ก็ผ่านตาพวกเราไปชุดหนึ่งแล้ว เพราะเท่าที่ผู้เขียนได้ลองใช้งานมาส่วนใหญ่ก็ใช้อยู่แค่นี้ เพราะตัวพัฒนาข้อมูลส่วนใหญ่ก็มีภาษาของตัวเองรองรับในส่วนที่การเรียกใช้ SQL อาจทำไม่สะดวกอยู่แล้ว แต่ถึงอย่างไรผู้เขียนก็ไม่อาจละเลยส่วนที่สำคัญอีกสองจุด ถ้าใครยังคิดไม่ออก ลองอ่านต่อไปก็จะถึงบางอ้อ

จริงอยู่การใช้ SQL มีความสะดวกเหลือหลายทีเดียวในดึงข้อมูลใน TABLE มาแสดงในรูปแบบต่าง ๆ ผลลัพธ์ที่ออกมาถ้าสังเกตให้ดีจะมีอยู่ 4 อย่างคือ เป็นตาราง เป็นเคอร์เซอร์ เป็นเท็กซ์ไฟล์ หรือไม่ก็

ออกมาทางเครื่องพิมพ์ คู่มือในตัวพัฒนาโปรแกรมส่วนใหญ่จะแสดงมาในรูปนี้ แต่ถ้าเราจะดึงผลลัพธ์มาจาก SQL โดยตรงเช่นค่าการคำนวณ การแจงนับ มาเป็นตัวแปรในโปรแกรมของเราเลยจะทำได้หรือไม่ (เนื่องจากผู้เขียนเคยงมหาอยู่ใน DELPHI ช่วงหนึ่งด้วยความงุนงง)

ทำได้ครับ (แต่แปลกที่ไม่ค่อยเจอใน Help) นั่นคือ SELECT INTO ซึ่งไวยากรณ์ก็คล้าย ๆ กับ SELECT นั่นแหละ

ตัวอย่าง เราจะดึงผลรวมยอดขายของนายสมชาย จากตาราง invoice มาใส่ในตัวแปร total

```
SELECT SUM(valueprice) INTO :total
FROM invoice
WHERE repcode='002'
```

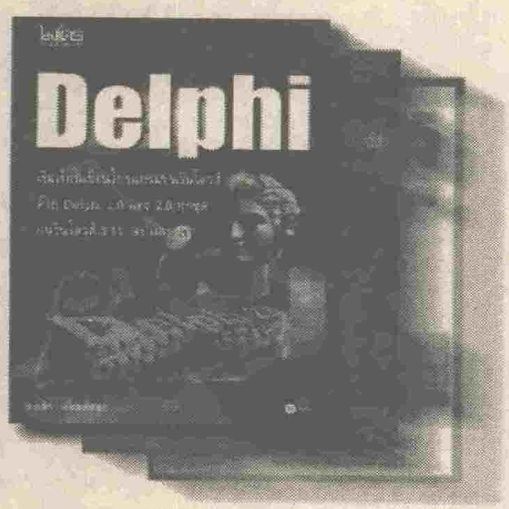
...จุดนี้ผ่านไปได้

ลองตั้งคำถามใหม่ ถ้าเราต้องการดูยอดขายของคนอื่น ๆ เช่น นายจำลอง นายเพียร ฯลฯ เรามิต้องมานั่งเขียน SQL ใหม่ทุกครั้งไปหรือ? นั่นคือจุดสำคัญอีกจุดหนึ่งคือทำอย่างไรจะให้ประโยค SQL ของเราสามารถรับค่าตัวแปรจากภายนอกได้ (บรรดาสาวก Ms.ACCESS คงยิ้มกริ่มเพราะหมูมาก) แต่สำหรับตัวพัฒนาอื่น ๆ ก็หมูเช่นกัน แต่เป็นหมูที่อยู่ตามป่า แถมมีเขี้ยวตันเสียด้วยสิ...

ลองทำแล้ว สรุปว่า ตัวพัฒนา 5 ตัวก็มีวิธีทำเกือบ 10 อย่าง แทบไม่เหมือนกันเลย ดังนั้นคราวหน้าตั้งใจว่าจะขอเสนอ **การใช้ SQL แบบรับตัวแปร (พารามิเตอร์)** โดยจะเจาะไปในตัวพัฒนาระบบฐานข้อมูลแต่ละตัว ว่ามีวิธีการทำได้อย่างไร จะพยายามทำในตัวพัฒนาข้อมูลให้มากตัวที่สุด ทั้งนี้เพื่อแฟน ๆ ซึ่งหลายจิตหลายใจใช้กันหลายภาษาจะได้ประโยชน์โดยพร้อมเพรียงกัน แค่คิดจะเขียนก็มันส์แล้วครับ

phasitphattaramoon@thaimail.com
jokeuse@health2.moph.go.th

แลกฟรี!



หนังสือคู่มือการใช้งานโปรแกรม 1. Delphi 2. Microsoft Windows 95 **อย่างง่าย และ 3. เทคนิคการเดินสายสัญญาณเครือข่ายคอมพิวเตอร์** กำลังรอให้คุณเป็นเจ้าของ

ง่าย ๆ เพียงแต่คุณเขียนเทคนิคเกี่ยวกับการใช้คอมพิวเตอร์ ที่สามารถนำมาประยุกต์ใช้ในปัจจุบัน เพื่อเพิ่มประสิทธิภาพในการทำงานให้กับฮาร์ดแวร์ อุปกรณ์ต่อพ่วง ซอฟต์แวร์ อินเทอร์เน็ต หรืออาจจะเป็นเทคนิคหรือเรื่องราวที่น่าสนใจอะไรก็ได้ที่เกี่ยวข้องกับคอมพิวเตอร์ โดยแต่ละเรื่องนั้น เมื่อพิมพ์ด้วยฟอนต์ขนาดปกติแล้วจะต้องมีความยาวไม่ต่ำกว่า 2/3 หน้ากระดาษ A4

เมื่อนึกออกแล้วก็ให้รีบเขียนส่งมายังที่อยู่ด้านล่าง หรือจะแนบไฟล์มากับอีเมล์ ส่งมาที่ june@se-ed.com (ในช่อง Subject ให้ระบุชื่อหนังสือที่ต้องการว่า 1. Delphi 2. MS Windows 95 หรือ 3. Wire) **เรื่องที่ผ่านการพิจารณาจะจัดลงตีพิมพ์ในนิตยสารไมโครคอมพิวเตอร์ คุณจะได้รับหนังสือคู่มือการใช้งานโปรแกรม Delphi, Microsoft Windows 95 อย่างง่าย หรือ เทคนิคการเดินสายสัญญาณเครือข่ายคอมพิวเตอร์ (1 เรื่องต่อ 1 เล่ม หากต้องการเล่มใดโปรดระบุ)**

กองบรรณาธิการนิตยสารไมโครคอมพิวเตอร์ **บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)** 800/43-45 ซอยตระกูลสุข ถนนอโศก-ดินแดง ดินแดง กรุงเทพฯ 10400 *(วงเล็บมุมซองว่า เทคนิคการใช้งาน)*